Manual de uso

Nota: Los apartados 3 y 5 de este manual se corresponden con los anexos 1 y 2 de la memoria de este PFG.

1 Instalación de PAPI

Para utilizar Papify, es necesario tener instalado PAPI (*Performance Application Programming Interface*) en el sistema. Las siguientes instrucciones asumen que se está utilizando una plataforma con Ubuntu u otra distribución de Linux derivada de Debian. Para otras plataformas, revisar la documentación de PAPI.

1. Para obtener y compilar PAPI es necesario git y gfortran:

   ```
   $ sudo apt-get install git

   $ sudo apt-get install gfortran
   ```

2. Obtener y compilar PAPI:

   ```
   $ git clone https://icl.cs.utk.edu/git/papi.git

   $ cd papi

   $ cd src

   $ ./configure

   $ make
   ```

3. Instalar:

   ```
   $ sudo make install-all
   ```

2 Instalación versión de desarrollo de ORCC

El último *release* de ORCC, de diciembre 2014, tiene una versión de *Papify* que no es compatible con *Papify-Viewer*. Si se desea generar una salida de Papify compatible con *Papify-Viewer*, es necesario utilizar la versión de desarrollo. De lo contrario, en este CD se incluyen salidas de prueba para probar con *Papify-Viewer* (ver `/Material Adicional/Ejemplos papi-output`).

Para instalar la versión de desarrollo de ORCC, seguir los siguientes pasos:

1. Descargar Eclispe 4.4 (Luna) *for Java and DSL Developers*.
2. Una vez en Eclipse, instalar Graphiti SDK: Menú "Help" > "Install New Software...". En la ventana que se abre, en “Work with” seleccionar “Luna”. Cuando termine de cargar, escribir en la barra de búsqueda “Graphiti SDK”, seleccionar “Graphiti SDK” y seguir las instrucciones en pantalla para terminar su instalación.
3. Copiar el código fuente de ORCC al disco (en este CD: `/Codigos fuente/orcc` o bien descargar de GitHub)
4. En Eclipse, importar la carpeta `/orcc/eclipse/plugins/`: File > Import > General > Existing projects into Workspace. En la ventana que se abre, seleccionar la carpeta plugins y click en *Finish*.
5. Abrir el proyecto net.sf.orcc.cal, navegar a `src/net.sf.orcc.cal`, click derecho sobre “GenerateCal.mwe2” > “Run As MWE2 Workflow”.
6. Cuando en la terminal de Eclipse se pregunta si se quiere descargar el “Antlr parser generator”, escribir “y” y presionar Enter.
7. Finalmente, cuando el paso anterior haya finalizado (puede demorar unos segundos), abrir el proyecto “net.sf.orcc.core”, click derecho sobre "plugin.xml" > “Run as” > “Eclipse Application”. Se abrirá una nueva instancia de Eclipse con la versión de desarrollo de ORCC, que incluye *Papify* compatible con *Papify-Viewer*.

Estos pasos serán necesarios hasta que salga el próximo *release* de ORRC. Los mismos también se encuentran disponibles en la página de ORCC.

Para instalar el *release* más reciente de ORCC, seguir los pasos del sitio oficial.

3 Manual de uso de *Papify*

En este anexo se presenta el manual de uso de *Papify*. Nótese que se parte de que el sistema donde se utiliza *Papify* tiene instalado Eclipse con ORCC y además cuenta con PAPI. La herramienta *Papify* en si no requiere ninguna instalación adicional, ya que viene incluida con ORCC.

3.1 Activar *Papify*

Para activar *Papify*, es necesario seguir el proceso normal de compilación de ORCC: sobre la red xdf que se vaya a compilar, *click* derecho -> *Run As* -> *Run Configurations*.

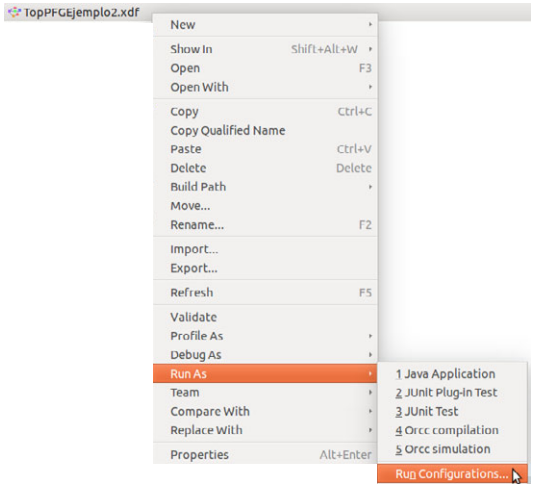

Fig. 3.1. Selección Run Configurations

En la ventana *Run Configurations*, seleccionar el *backend* C.

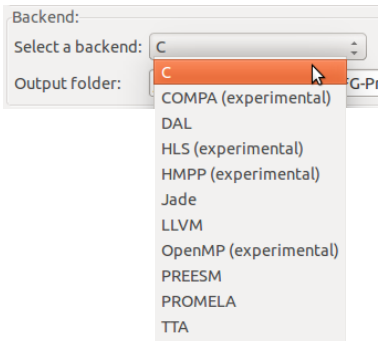

Fig. 3.2. Selección backend C.

Dentro de las opciones del *backend*, Seleccionar la última opción: "*Papify: profile actors using* PAPI". Opcionalmente, seleccionar "*Activate multiplex*" en caso de que la cantidad de contadores PMC disponibles en la plataforma sea menor que la cantidad de eventos configurada.

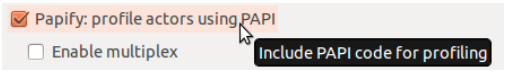

Fig. 3.3. Selección opción Papify

Posteriormente, *click* en *Apply* y luego en *Run* para generar el código fuente con PAPI incluido en los actores y acciones configurados.

3.2 Evaluar actores

Para evaluar actores completos (todas sus acciones) solo basta con añadir una anotación justo antes de la primera línea del actor con el formato "`@papify([evento`$_1$`], [evento`$_2$`], … ,[evento`$_n$`])`", donde cada elemento `evento`$_i$ es un evento de PAPI. Al menos un evento debe ser añadido en esta anotación.

Por ejemplo, si se quiere medir las prestaciones del actor *Rotate* del programa Rotador con los eventos "*PAPI_TOT_INS*" y "*PAPI_L1_DCM*", arriba de la declaración del actor habrá que agregar "`@papify(PAPI_TOT_INS, PAPI_L1_DCM)`", como se muestra en la figura 3.4**Error! Reference source not found.**.

```
1 @papify(PAPI_TOT_INS, PAPI_L1_DCM)
2 actor Rotated90()
3 uint(size=8) Y ==> uint(size=8) YRot :
4 /*(...)*/
```

Fig. 3.4. Ejemplo de cómo utilizar anotaciones para añadir Papify en la especificación de un actor

Para conocer los eventos disponibles en un sistema con PAPI instalado, una buena opción es utilizar el comando `papi_avail`. Otro comando útil a la hora de añadir eventos es `papi_event_chooser`. Este comando permite averiguar qué eventos son "asociables" entre sí. Por limitaciones de hardware, es posible que determinados eventos no sean compatibles entre sí en la misma ejecución. Por ejemplo, el comando "`papi_event_chooser PRESET PAPI_TOT_INS`" dará una lista de eventos que es posible mezclar con "*PAPI_TOT_INS*". Del mismo modo, "`papi_event_chooser PRESET PAPI_TOT_INS PAPI_L1_DCM`" dará como resultado la lista de eventos compatibles con esos dos eventos.

3.3 Evaluar acciones

Si solo interesa evaluar una o más acciones en particular de un actor, pero no todas las acciones, es posible agregando anotaciones "`@papify`" a aquellas acciones que requieran ser medidas. Además, habrá que realizar el procedimiento mencionado para evaluar actores en el apartado anterior, de modo que las acciones con la anotación "`@papify`" serán configuradas con los eventos añadidos en la anotación del actor.

Por ejemplo, en la figura 3.5 se realiza medida de prestaciones únicamente a la acción *Image_load* del actor *Rotate*, con los eventos "PAPI_TOT_INS" y "PAPI_L1_DCM".

```
1  @papify(PAPI_TOT_INS, PAPI_L1_DCM)
2  actor Rotated90()
3  uint(size=8) Y ==> uint(size=8) YRot :
4
5  /* (...) */
6
7   @papify
8   Image.load: action Y:[ pixel ] ==>
9   guard LoadUnload=true
10
11  /* (...) */
```

Fig. 3.5. Realizar medidas sobre una única acción

3.4 Carpeta de salida

Al terminar la ejecución de un programa configurado con *Papify*, aparecerá una nueva carpeta llamada "*papi-output*". Si se sigue el proceso normal de compilación de la aplicación y se mantiene el ejecutable en su lugar por defecto, esta carpeta aparecerá dentro del directorio */bin/* de la ruta de directorios creada por ORCC.

Dentro de la carpeta "*papi-output*" estarán localizados los ficheros de salida de cada actor configurado con *Papify*.

3.5 Evaluar un descodificador de vídeo

Descodificar vídeo es normalmente un proceso iterativo. Una buena idea al evaluar descodificadores es limitar la cantidad de información a la descodificación de un único cuadro o a lo sumo dos o tres.

Para conseguir esto para un programa generado con ORCC y configurado con *Papify*, una vez compilado, ejecutar la aplicación con la opción de línea de comando "*-f* 1". Esto limitará la aplicación a la descodificación de un único cuadro, reduciendo notablemente la cantidad de datos generados.

4 Ejecutar Papify-Viewer

1. Para ejecutar Papify-Viewer, copiar la siguiente carpeta del CD al disco:

   `/Codigos fuente/papify_viewer/application.linuxYY`

   (donde `YY`, seleccionar 32 o 64 (bits) de acuerdo a su instalación)

2. Abrir un terminal (en Ubuntu: ctrl+alt+T), navegar hasta el directorio recién copiado y ejecutar:

   ```
   $ chmod +x papify_viewer
   ```

   ```
   $ ./papify_viewer
   ```

Si se desea utilizar los ejemplos incluídos en `/Material Adicional/Ejemplos papi-output/` dentro de este CD, es recomendable también copiar los ejemplos al disco para acelerar la ejecución del visualizador.

5 Manual de uso de *Papify-Viewer*

En este anexo se presenta el manual de uso de la herramienta *Papify-Viewer*. Para utilizar *Papify-Viewer* es necesario previamente haber generado al menos un fichero CSV de salida con *Papify*.

5.1 Interfaz

Al abrir *Papify-Viewer*, la primera pantalla que se presenta es la interfaz sin ninguna información, como se muestra en la figura 5.1. Arriba a la izquierda se tiene la barra de herramientas. A la derecha de la barra de herramientas se muestra la barra de información, utilizada para mostrar información sobre procesos que se están llevando a cabo, información sobre las herramientas o bien información de ayuda. El rectángulo de abajo es donde se dibuja la visualización temporal de la aplicación una vez cargados los ficheros CSV de *Papify*. Las pestañas "*Partitions*" y "*Actors*" permiten cambiar entre las vistas de particiones (o núcleos) y actores, y se activan según la información cargada en el programa.

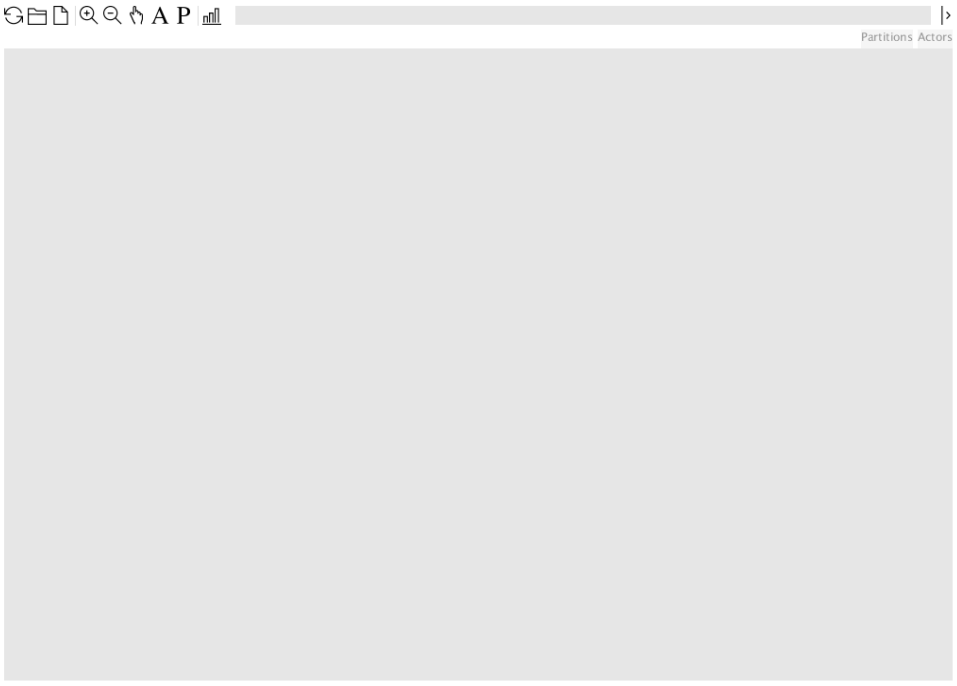

Fig. 5.1. Interfaz de Papify-Viewer

5.2 Herramientas

La barra de herramientas consta de nueve íconos, como se puede observar en la figura 5.2.

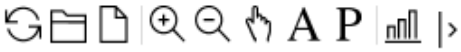

Fig. 5.2. Barra de herramientas

De izquierda a derecha, estos son: refrescar, cargar carpeta *papi-output*, cargar fichero de mapeo XCF, aumento, reiniciar aumento, herramienta puntero, herramienta de actores, herramienta de particiones y estadísticas globales.

- **Refrescar**: Se utiliza para recargar la visualización temporal. Útil en caso de que la visualización contenga errores o bien para aplicar cambios en los parámetros de visualización.

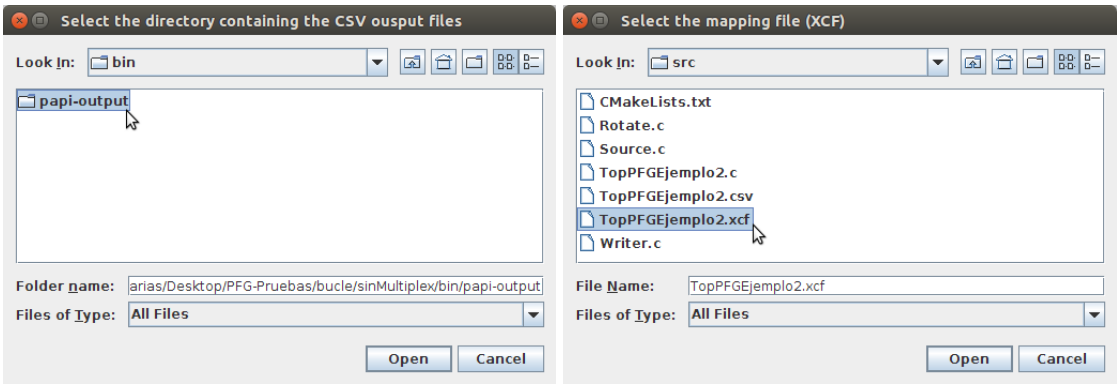

Fig. 5.3. Carga de carpeta de salida de Papify (izq.) y carga de fichero de mapeo xcf (der.)

- **Cargar salida de *Papify* y cargar fichero de mapeo**: El primero permite cargar la carpeta de salida generada por *Papify*, *papi-output* (figura 5.3**Error! Reference source not found.**, izquierda). Una vez cargada, se representa la línea temporal de la ejecución de la aplicación como se muestra en la figura 5.4. Al cargar el fichero XCF de mapeo (figura 5.3, derecha), se representa una visualización similar, sólo que agrupando los actores por núcleo.

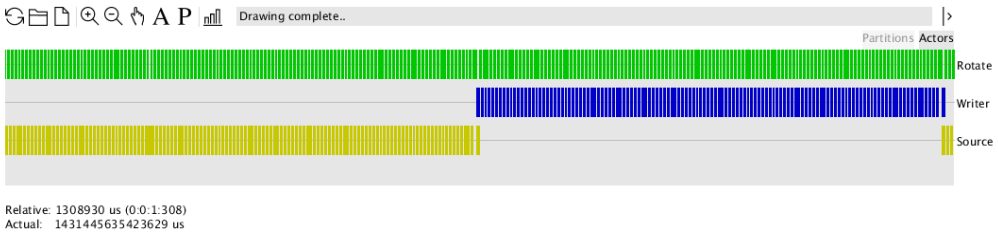

Fig. 5.4. Actores cargados, visualización temporal

- **Aumento**: El ícono de la lupa con el símbolo "+" adentro permite seleccionar una zona temporal, como se muestra en la figura 5.5, arriba. Una vez seleccionada la zona, se aplica el aumento, reduciendo el ancho del eje de tiempos. Esta operación puede repetirse tantas veces como se desee hasta alcanzar el límite de resolución temporal. El ícono de la lupa con el símbolo

"-" reinicia la vista a la inicial, ampliando el eje de tiempos desde el primer evento registrado hasta el último.

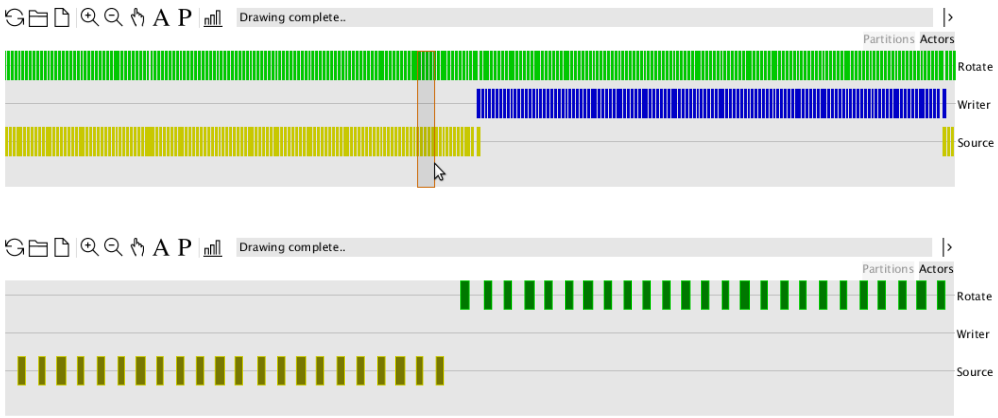

Fig. 5.5. Utilización de la herramienta de aumento (arriba) y aumento aplicado (abajo)

- **Herramienta puntero**: La herramienta puntero permite obtener información adicional sobre un punto específico de la ejecución. En la vista de actores, permite identificar qué acción en particular se ha ejecutado dentro del actor seleccionado en un momento dado, como se muestra en la figura 5.6. De forma similar, en la vista de particiones, permite identificar qué actor se ha ejecutado en un momento determinado en la partición seleccionada.

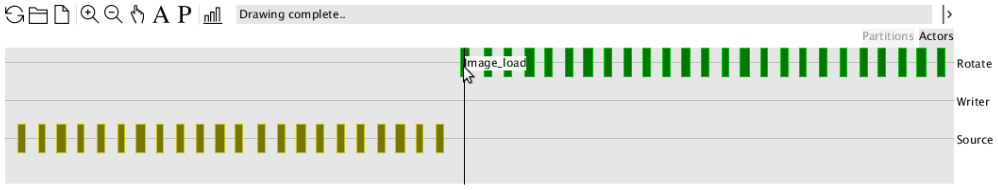

Fig. 5.6. Herramienta puntero

- **Herramienta de actores**: La herramienta de actores lista todos los actores medidos de la aplicación, como se muestra en la figura 5.7. Permite ocultar actores en la vista temporal por si se desea centrarse en algunos actores en particular. Además, permite abrir la ventana de estadísticas de actor, como la que se muestra en la figura 5.8**Error! Reference source not found.**, dividiendo el recuento de eventos por acción. En esta ventana es posible circular entre los eventos medidos, cambiar la escala entre logarítmica y lineal o ver el promedio de evento de cada acción. Al pasar el puntero por encima de cada barra es posible visualizar información adicional.

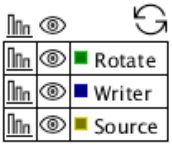

Fig. 5.7. Herramienta de actores

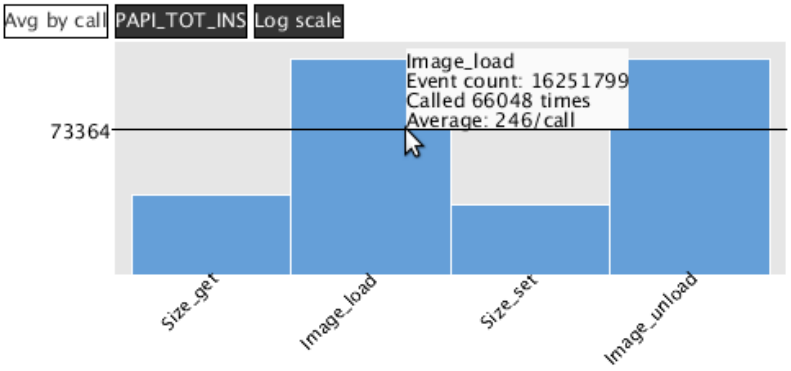

Fig. 5.8. Estadísticas del actor

- **Herramienta de particiones**: Análoga a la herramienta de actores. Permite ocultar particiones en la vista temporal y mostrar estadísticas de partición (figura 5.9).

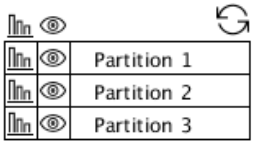

Fig. 5.9. Ventana de particiones

- **Estadísticas globales**: Esta opción permite abrir la ventana de estadísticas globales por actor, como se muestra en la figura 5.10. En esta vista es posible observar el recuento total de eventos de todas las acciones de cada actor.

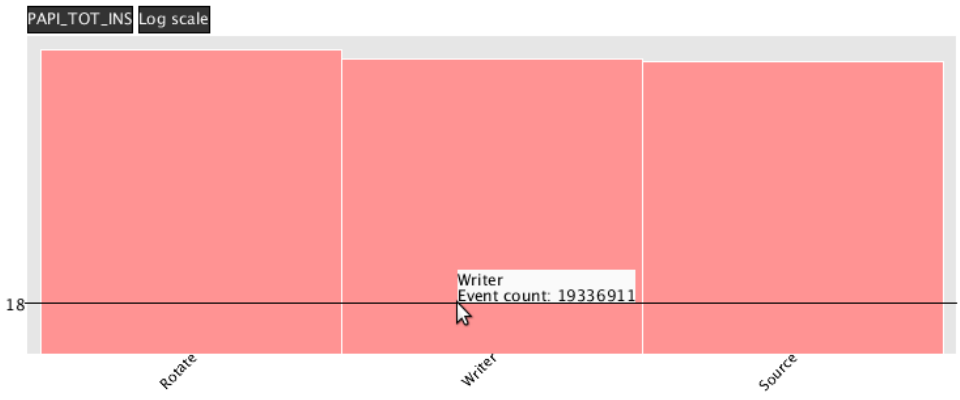

Fig. 5.10. Estadísticas globales por actor

6 Ejecutar programa Rotador

El programa Rotador es el utilizado como ejemplo a lo largo de la memoria de este PFG.

1. Copiar a disco la carpeta Rotador que se encuentra dentro de `/Codigos fuente` en este CD
2. Importar la carpeta recién copiada dentro de Eclipse: File > Import > General > Existing projects into Workspace. En la ventana que se abre, seleccionar la carpeta Rotador (en el disco!) y click en *Finish* (se importan los proyectos PFGEjemplo1 y System).
3. Seguir los pasos para generar el programa rotador instrumentado (ver el apartado 3.1) utilizando como red xdf el fichero `PFGEjemplo1/src/ec.upm.citsem.pfgejemplo1/TopPFGEjemplo2.xdf`.
4. Una vez traducido a código C, compilar el rotador instrumentado:

   En la carpeta `build`, dentro del directorio con el código generado:

   ```
   $ cmake ..

   $ make
   ```

5. Ejecutar el rotador:

   Tras colocar el fichero raw de entrada en la carpeta bin:

   ```
   $ ./TopPFGEjemplo2 -i ficheroDeEntrada.raw -w
   ficheroDeSalida.raw
   ```

En la carpeta `/Material adicional/Ejemplos imagenes raw/` hay imágenes raw de prueba.

7 Instalar libtiff y tiff2raw

Aunque estas herramientas no tienen nada que ver con el proyecto, pueden resultar útiles si se desea probar el programa rotador.

Instalar libtiff:

1. Copiar tiff-4.0.3.zip al disco. En este CD se encuentra en `/Material adicional/libtiff & tiff2raw/`
2. Descomrpimir tiff-4.0.3.zip.
3. Compilar e instalar:

```
$ cd tiff-4.0.3

$ ./configure

$ make

$ sudo make install
```

Compilar tiff2raw:

1. Mover tiff2raw.c de la carpeta `/Material adicional/libtiff & tiff2raw/` al mismo directorio que se uso al instalar libtiff.
2. Compilar e instalar en /usr/bin/

```
$ gcc tiff2raw.c -Itiff-4.0.3/libtiff -L/usr/lib -o tiff2raw
-ltiff

$ sudo cp tiff2raw /usr/bin/
```

Para transformar de raw a tiff:

```
$ raw2tiff ficheroDeEntrada.raw ficheroDeSalida.tiff
```

Para transformar de tiff a raw:

```
$ tiff2raw ficheroDeEntrada.tiff ficheroDeSalida.raw
```